САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

КАФЕДРА ТЕХНОЛОГИИ ПРОГРАММИРОВАНИЯ

Назукин Дмитрий Евгеньевич

Выпускная квалификационная работа бакалавра

Система автоматического распознавания дорожных знаков

Направление 010300

Фундаментальная информатика и информационные технологии

Научный руководитель,
старший преподаватель
Малинин К.А.

Санкт-Петербург

2016

Содержание

Введение

В современном мире обязательным атрибутом в организации дорожного движения являются дорожные знаки. Они информируют водителей об опасных участках дороги, указывают направление движения, запрещают или дают право проезда, обязывают снизить скорость, а также выполняют множество других полезных задач.

С каждым годом число водителей постоянно увеличивается. Так по данным аналитики компании “Автостат Инфо” [1] количество легковых автомобилей в нашей стране по состоянию на 1 января 2016 года достигло 40 млн 629,2 тыс. Относительно 1 января 2015 года емкость автопарка выросла на 1 млн 279,9 тыс. В связи с этим необходимо, чтобы дорожное движение стало более безопасным. Для этой цели разрабатываются продвинутые системы помощи водителю “Advanced Driving Assistance Systems” (ADAS), которые постепенно встраиваются в некоторые современные высокотехнологичные серии автомобилей.

Система автоматического распознавания дорожных знаков является важнейшей частью ADAS. Она призвана уведомлять водителя о наличии дорожных знаков на дороге. Система может помочь водителю придерживаться установленного на участке дороги скоростного ограничения, соблюдать ограничения на проезд, обгон и т.д. Подобные системы впервые появились в конце 2008 года в автомобилях BMW 7-Series, а спустя год и в Mercedes-Benz S-Class. Эти системы умели распознавать только знаки ограничения скорости. Позже их встроили в такие серии как Volkswagen Phaeton и Opel Astra с дополнительной функцией распознавания знака “обгон запрещен” [2]. В общем, все ведущие мировые производители автомобилей стараются встроить подобные технологии в свои продукты. Однако большинство людей не может позволить себе цену подобных автомобилей и,

как следствие, не может воспользоваться данной технологией. С другой стороны, на сегодняшний день практически у каждого есть смартфон с камерой. К тому же вычислительные мощности телефонов с каждым днем увеличиваются и уже приближаются к компьютерным. Поэтому наличие системы автоматического распознавания дорожных знаков в телефоне может решить эту проблему.

Задачей работы является осуществление идеи по разработке системы автоматического распознавания запрещающих и предупреждающих дорожных знаков и знака “Пешеходный переход”, способной работать на платформе Android в режиме реального времени. Были выбраны эти знаки, потому что именно они обозначают наиболее важную информацию для водителя, и их нарушение приводит к наиболее опасным ситуациям на дороге.

Система автоматического распознавания дорожных знаков представляет собой систему, как правило, состоящую из двух основных этапов:

1. Локализация дорожных знаков.
2. Классификация дорожного знака.

В зависимости от реализации эти этапы сами по себе тоже состоят из подэтапов. В данной работе локализация знаков реализована методом Виолы-Джонса. Задачу классификации выполняет обученная свёрточная нейронная сеть.

Постановка задачи

Целью данной работы является разработка системы автоматического распознавания запрещающих и предупреждающих дорожных знаков и знака “Пешеходный переход” на платформе Android.

Для достижения этой цели будут решены следующие задачи:

1. Исследовать и изучить научную литературу и основные подходы по рассматриваемой теме.
2. Изучить необходимые инструменты и технологии.
3. Подготовить базу дорожных знаков.
4. Реализовать этап локализации дорожных знаков.
5. Реализовать этап классификации дорожных знаков.
6. Реализовать систему на платформе Android.
7. Проанализировать результаты работы системы.

Глава 1. Теоретическая часть

В данной главе будут исследованы этапы локализации и классификации дорожных знаков. Существующие методы решения задачи локализации можно разделить на 3 категории:

1. Методы, опирающиеся на цветовые признаки знаков.
2. Методы, опирающиеся на форму знака.
3. Методы, основанные на машинном обучении.

В данной работе был выбран метод, основанный на машинном обучении — метод Виолы-Джонса (построение каскада классификаторов). Критериями выбора стали:

- высокие показатели точности и скорости работы [3];
- наличие реализации метода в библиотеке OpenCV.

При решении задачи классификации изображений наибольшие помехи, влияющие на результат, вносят аффинные и проекционные искажения, которые возникают в связи с изменением угла регистрации, изменением масштаба, а также искажения, связанные с плохими погодными условиями. В 90-х годах прошлого века Ян Лекун разработал мощный инструмент для распознавания изображений — свёрточные нейронные сети. Их успешно применяли к задачам распознавания рукописных букв [4], лиц [5] и многим другим, где они показали хорошую устойчивость к искажениям подобного рода и высокую скорость работы. В связи с этим было решено использовать именно этот аппарат для распознавания дорожных знаков.

Рассмотрим основные теоретические аспекты используемых методов.

1.1. Метод Виолы-Джонса

Исследование Пола Виолы и Майкла Джонса [3] имеет огромную значимость в области компьютерного зрения. Виола и Джонс разработали

алгоритм, способный обнаруживать объекты очень надежно и достаточно быстро, чтобы работать в режиме реального времени. Хотя первоначально метод предназначался для решения задачи обнаружения лиц, различные исследователи успешно применили его к распознаванию других классов объектов.

Детектор Виолы и Джонса представляет собой каскад классификаторов. Он сочетает в себе следующие концепции:

- используются признаки Хаара;
- изображения представляются в интегральном виде;
- применяется бустинг.

Признак Хаара представляет собой прямоугольный примитив. Изначально авторами было предложено использовать только примитивы четырех типов (рис. 1а), но также, например в библиотеке OpenCV, используются и дополнительные (рис. 1б).

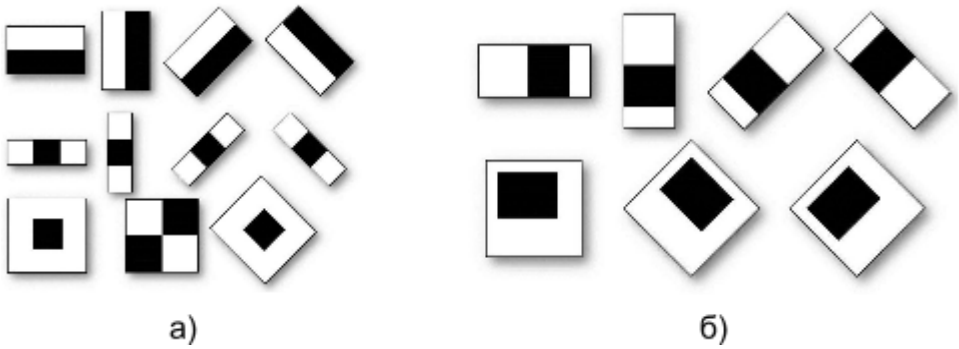

Рис. 1. Примитивы Хаара: а) стандартные, б) дополнительные

Значение признака считается по формуле:

$$F = X - Y ,$$

где X — сумма значений пикселей закрываемых светлой частью признака, а Y — сумма значений пикселей закрываемых темной частью признака.

Каждый признак работает в паре с пороговым значением, а решение признака определяется путем сравнения его значения с пороговым. Виола и Джонс предложили очень быстрый способ вычисления значений признаков,

который использует интегральное представление изображения. Интегральное представление изображения – это матрица, имеющая размеры как у исходного изображения, значение элементов в которой определяются как сумма интенсивностей пикселей находящихся левее и выше (рис. 2).

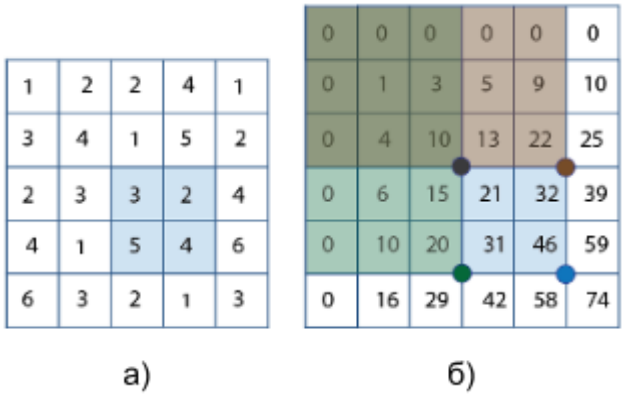

Рис. 2. Представление изображения: а) растровое, б) интегральное

По такой матрице сумма значений пикселей в произвольном прямоугольнике вычисляется за постоянное время. Пусть есть интересующий нас прямоугольник $ABCD$ (рис. 3):

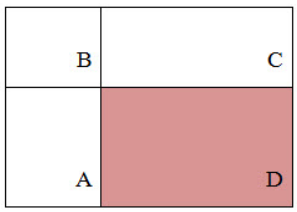

Рис. 3. Прямоугольник *ABCD*.

Из рисунка понятно, что сумму внутри прямоугольника можно выразить через суммы и разности смежных прямоугольников по следующей формуле:

$$S(ABCD) = L(D) + L(B) - L(A) - L(C),$$

где $L(A)$ — значение интегрального представления в точке A.

Бустинг — комплекс методов, способствующих повышению точности аналитических моделей. “Слабые” модели не позволяют классифицировать объекты, делают в работе большое количество ошибок. Поэтому бустинг (от англ. *boosting* — повышение, усиление, улучшение) призван усилить слабые

модели путем последовательного построения их композиций, чтобы каждая последующая модель исправляла ошибки предыдущей.

AdaBoost (adaptive boosting) — это метод, объединяющий ряд слабых классификаторов (признаков Хаара) в один сильный. *AdaBoost* присваивает веса признакам на основе их качества, и в результате сильный классификатор представляет собой линейную комбинацию слабых классификаторов с соответствующими весами. Метод Виолы и Джонса объединяет несколько сильных классификаторов построенных методом *AdaBoost* в каскад. Для обучения каскада строятся положительная и ложная выборки. Классификатор на первой ступени подбирается таким образом, чтобы с малым количеством примитивов отбрасывать большое количество ложных объектов, сохраняя при этом почти все положительные объекты обучающего множества. Для каждого последующего этапа количество примитивов увеличивается, ложно-положительные срабатывания предыдущего этапа обозначаются как негативные элементы выборки и обучение продолжается. Следовательно, последующие этапы проходят обучение, чтобы исправить ошибки предыдущих, при этом сохраняя высокую точность на истинных элементах выборки. Таким образом, использование каскада позволяет быстро отбрасывать большую часть ложных объектов на ранних стадиях, что сильно уменьшает количество вычислений на следующих стадиях. Пример каскада представлен на рисунке 4.

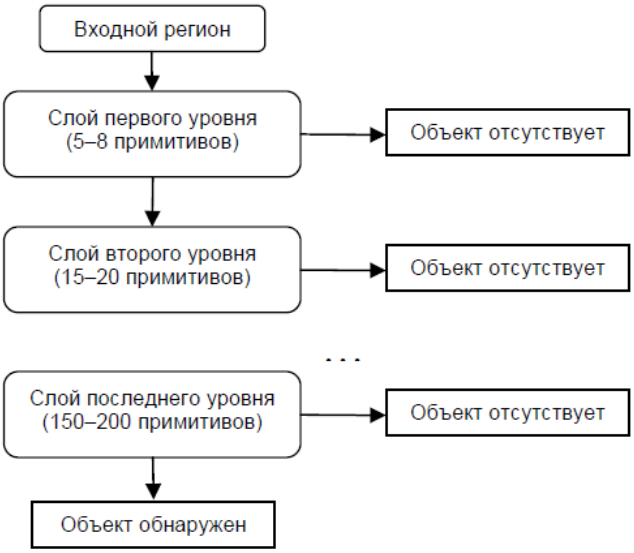

Рис. 4. Пример каскада классификаторов

Процесс детектирования осуществляется путем скольжения окна обнаружения по всему изображению. По каждому окну вычисляется решение каскада. В случае положительного ответа считается что внутри окна находится искомый объект. После завершения одного прохода по изображению, размер окна увеличивается (в OpenCV по умолчанию в 1,2 раза, а это означает, что масштаб окна будет увеличен на 20%). Размер окна увеличивается пока не будет достигнут некоторый заранее определенный размер. Меньший процент увеличения, улучшает показатель обнаружения, но увеличивает общее время обработки.

1.2. Свёрточные нейронные сети

Свёрточные нейронные сети включают три основные парадигмы:

1. *Локальное восприятие.* Данная идея означает, что на вход каждого нейрона подается не все изображение (или выходы предыдущего слоя), а только некоторая его область, причем на каждый нейрон своя. Это значительно сокращает объем вычислений и в то же время позволяет сохранить топологию изображения от слоя к слою. При таком подходе

каждый нейрон следующего слоя получает информацию не от всего предыдущего слоя, а только от его части, которая описывается матрицей (обычно ее выбирают размером 5×5). “Сканирование” нейроном не всего изображения, а только его части позволяет уловить особенности изображения, что значительно увеличивает точность распознавания.

2. *Разделяемые веса.* Данная концепция позволяет для большого количества связей использовать относительно небольшое число обучаемых параметров — весовых коэффициентов. Достигается это за счет того, что все нейроны одной карты имеют одинаковые весовые коэффициенты (матрицу весов). Поэтому при обучении сети для целой карты происходит корректировка только матрицы весов. В результате чего обучение сети проходит намного быстрее чем у обычного персептрона, у которого при обучении настраивается каждая связь между нейронами соседних слоев. Например, рассмотрим количество настраиваемых параметров между двумя картами 14×14 и 10×10 с матрицей весов размера 5×5. В свёрточной сети это будет количество элементов матрицы весов, т.е. 25 параметров, в персептроне же это будет количество всех связей между картами, т.е. $14 \cdot 14 \cdot 10 \cdot 10 = 19600$ параметров. При этом от такого допущения точность распознавания не уменьшается.
3. *Субдискретизация.* Суть данной идеи состоит в уменьшении пространственной размерности изображения. Существуют слои с одноименным названием, которые не анализируют изображение, а только уменьшают его размер, не теряя уже выделенные признаки. Этот подход позволяет добиться частичной инвариантности к масштабу.

1.2.1. Архитектура

Структура свёрточных нейронных сетей включает в себя чередование свёрточных и субдискретизирующих (подвыборочных) слоев вначале и наличие нескольких полносвязных слоев на выходе (рис. 5).

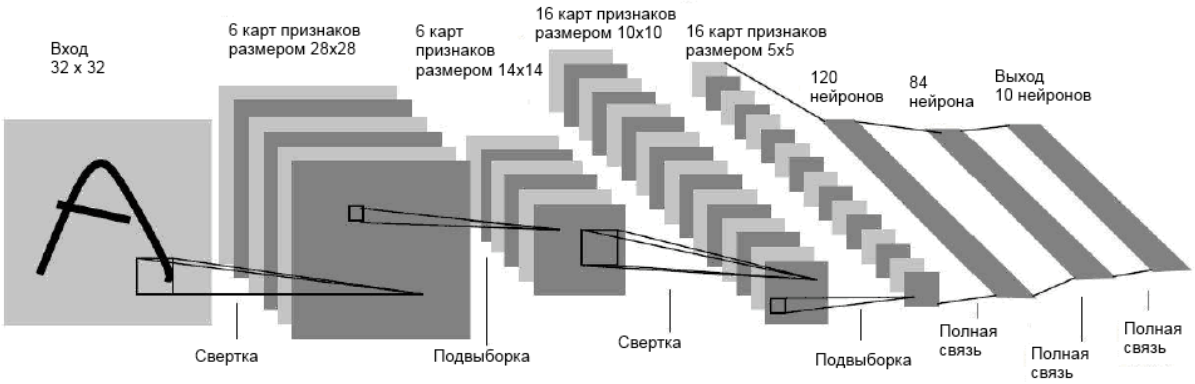

Рис. 5. Пример структуры свёрточной нейронной сети

1) Свёрточный слой (*convolutional layer*).

   Свое название слой обрел благодаря операции свёртки. Суть этого слоя заключается в следующем: ограниченная матрица весов двигается по обрабатываемому слою (изначально по изображению), поэлементно умножается на фрагмент слоя, результат суммируется и подается на вход функции активации соответствующего нейрона свёрточного слоя. Матрица весов, она же набор весов или ядро свёртки, как бы “графически кодирует” какой-либо признак, например, наличие угла или линии. По принципу разделяемых весов каждое ядро свёртки формирует отдельный экземпляр карты признаков, а т.к. обычно используется несколько ядер нейронная сеть получается многомерной (на одном слое много независимых карт). Важный факт, что при обработке матрицей весов слоя ее смещают каждый раз не на полный шаг (размер матрицы), а на небольшой. Например, при размерности матрицы весов 5×5 чтобы не

“пропустить” искомый признак, ее, как правило, сдвигают на один или два нейрона вместо пяти.

2) Субдискретизирующий слой (*subsampling layer*).

   Операция субдискретизации выполняет функцию уменьшения размерности сформированных карт признаков. Ее суть заключается в следующем: из небольшой квадратной области нейронов карты признаков выбирается максимальный и принимается за один нейрон карты признаков этого слоя. Реже используют операцию нахождения среднего значения. Данный слой не только ускоряет дальнейшие вычисления, но и позволяет спроектировать сеть более инвариантную к масштабу входного изображения.

3) Полносвязный слой (*full connection layer*).

   Этот слой представляет собой обычный персептрон, хорошо зарекомендовавший себя в распознавании простых объектов.

1.2.2. Обучение сети

Для обучения свёрточных нейронных сетей используются алгоритмы обучения с учителем [6]. Сначала на вход сети подается образ и вычисляется выход. Сравнение фактического выхода с желаемым дает возможность изменять веса связей таким образом, чтобы сеть на следующем шаге могла выдавать более точный результат.

При обучении ошибка работы сети может быть посчитана разными способами, самый стандартный — это функция среднеквадратичной ошибки:

$$E^{p}(w) = \frac{1}{2}\sum (x_k - d_k)^2 , \qquad (1)$$

где E^p — это ошибка распознавания для *p-ой* обучающей пары, k — номер выходного нейрона, x_k — реальное значение выходного сигнала нейрона, d_k — ожидаемое значение.

Задача обучения состоит в настройке весов таким образом, чтобы для любой обучающей пары ошибка E^p была минимальной. Ошибка для всей обучающей выборки считается как среднее арифметическое по ошибкам для обучающих пар. Такую усредненную ошибку обозначим как E. В вопросе минимизации функции ошибки наилучшим образом себя зарекомендовали градиентные методы [6]. Рассмотрим их суть на примере простейшего одномерного случая (когда у нас всего один вес). Разложив функцию ошибки E в ряд Тейлора в окрестности текущего веса, получится следующее выражение:

$$E(w_k + p_k) = E(w_k) + p_k \frac{dE(w_k)}{dw} + \frac{1}{2} p_k^2 \frac{d^2E(w_k)}{dw^2} + \dots, \qquad (2)$$

где E — функция ошибки, w_k — текущее значение веса.

Для достижения минимума функции (2) требуется, чтобы $\frac{dE(w_k+p_k)}{dp_k} = 0$. Если в формуле (2) ограничиться только первыми тремя слагаемыми, то при выполнении соответствующего дифференцирования можно получить необходимое направление движения p_k:

$$\frac{dE(w_k+p_k)}{dp_k} = \frac{dE(w_k)}{dw} + p_k \frac{d^2E(w_k)}{dw^2} = 0$$

$$p_k = -\left(\frac{d^2E(w_k)}{dw^2}\right)^{-1} \frac{dE(w_k)}{dw} \qquad (3)$$

$$w_{k+1} = w_k + h_k \cdot p_k \qquad (4)$$

В процессе поиска минимального значения целевой функции направление вектора p_k и шаг h_k подбираются таким образом, чтобы для каждой очередной точки w_{k+1} выполнялось условие $E_{k+1} < E_k$. Поиск минимума продолжается пока значение функции ошибки E не упадет ниже

заранее заданного значения, либо пока не будет превышено количество итераций. Формула (1) позволяет вычислить производную ошибки по весам, находящимся в выходном слое. Для нахождения значения ошибки в скрытых слоях пользуются методом обратного распространения ошибки [6].

При многомерном случае (т.е. для матрицы весов) все аналогично, только первая производная превращается в градиент (вектор частных производных), а вторая производная превращается в Гессиан (матрицу вторых частных производных). И здесь есть два варианта: использовать или не использовать Гессиан. Если его опустить, то получится алгоритм наискорейшего градиентного спуска. Если же использовать, то его подсчет потребует большое количество вычислений, поэтому обычно Гессиан заменяют чем-то более простым. Например, один из наиболее известных и успешных методов — метод Левенберга-Марквардта заменяет Гессиан его аппроксимацией с помощью квадратного Якобиана [7].

В общем виде алгоритм обучения нейронной сети выглядит следующим образом:

1. Проверка оптимальности текущего значения весов w_k. Если оно отвечает условиям окончания обучения — завершение процесса, иначе шаг 2.
2. Определение вектора направления оптимизации p_k для w_k по формуле (3).
3. Выбор величины шага h_k в направлении p_k, при котором выполняется условие $E(w_k + h_k \cdot p_k) < E(w_{k+1})$.
4. Определение нового значения w_{k+1} по формуле (4) и возврат к шагу 1.

1.2.3. Функция активации

Одним из важнейших аспектов при обучении нейронных сетей является выбор функции активации, которая вносит в сеть нелинейность. От ее выбора

зависит и скорость обучения, и точность распознавания. Существует несколько наиболее популярных функций активаций:

- Сигмоида (*sigmoid*).

  Сигмоида математически задается следующей формулой:

$$f(x) = \frac{1}{1+e^{-x}}$$

  Ее график представлен на рисунке 6. Эта функция принимает на вход произвольное вещественное число, а на выходе выдает вещественное число из интервала $(0, 1)$. В частности большие отрицательные числа приближаются к нулю, а большие положительные — к единице. Исторически сигмоида пользовалась большой популярностью, потому что ее значение легко интерпретирует уровень активации нейрона: 0 — полное отсутствие активации, 1 — полностью насыщенная активация.

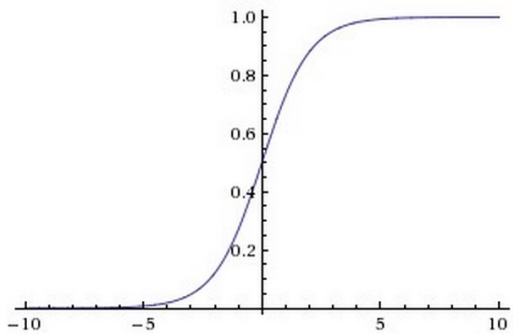

Рис. 6. График сигмоиды

  Однако на текущий момент сигмоида утратила свою былую популярность и используется очень редко, потому что она обладает серьезными недостатками:

  1. *Сигмоида склонна к "убийству" градиента (насыщению).* Этот недостаток заключается в том, что при стремлении функции к нулю или единице, ее градиент становится близок к нулю. При обратном распространении ошибки данный градиент умножается на градиент выхода. Следовательно, если градиент функции мал, он фактически обнуляет общий градиент. Как итог, при обучении

через такой нейрон сигнал практически не будет проходить к весам. Кроме того, если при начальной инициализации весов задать большие веса, большинство нейронов перейдут в состояние насыщения, в результате чего сеть будет плохо обучаться.

2. *Выход сигмоиды не центрирован относительно нуля.* Это свойство нежелательно, но не так критично как предыдущее. Оно плохо влияет на динамику градиентного спуска. Если в нейрон поступают всегда положительные значения (например, $x > 0$ поэлементно в $f = w^T x + b$), тогда в процессе обратного распространения ошибки все градиенты весов w будут либо положительны, либо отрицательны. Из-за этого веса могут нежелательно зигзагообразно обновляться. Тем не менее, можно заметить, что когда эти градиенты складываются по пакету данных окончательное обновление для весов может иметь переменный характер, уменьшая данную проблему.

- Гиперболический тангенс (*hyperbolic tangent, tanh*).

  Задается формулой:

$$f(x) = \frac{1}{1+e^{-2x}} - 1$$

  График представлен на рисунке 7. Эта функция принимает на вход произвольное вещественное число, а на выходе выдает вещественное число из интервала $(-1, 1)$. Подобно сигмоиде, гиперболический тангенс может перенасыщаться. Однако, здесь решена вторая проблема сигмоиды — выход центрирован относительно нуля. Поэтому более предпочтительно использовать гиперболический тангенс, а не сигмоиду.

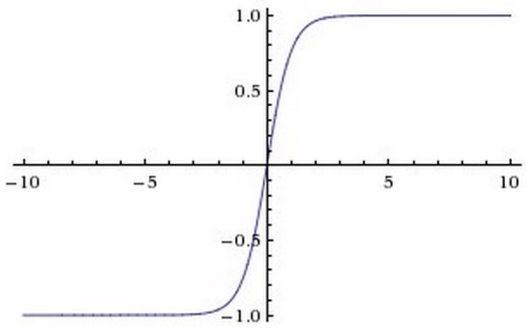

Рис. 7. График гиперболического тангенса

- Выпрямленная линейная функция активации (ReLU).

  Rectified Linear Unit (ReLU) функция стала очень популярной в последние несколько лет [8]. Она задается формулой (5), которая просто делает пороговый переход в нуле. Ее график представлен на рисунке 8.

$$f(x) = max(0, x) \tag{5}$$

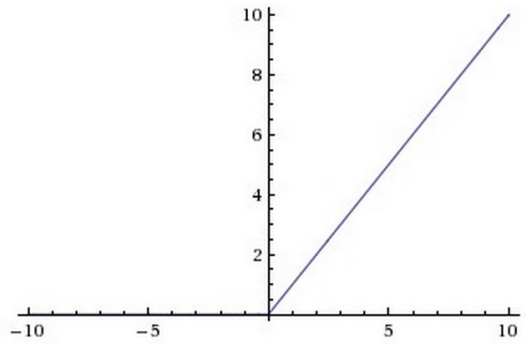

Рис.8. График ReLU функции

Положительные стороны:

1. По сравнению с сигмоидой и гиперболическим тангенсом, которые используют дорогостоящие операции (вычисление экспоненты, деление и т.д.), ReLU может быть реализована простым пороговым преобразованием. В то же время, ReLU не склонна к перенасыщению.
2. Было обнаружено, что эта функция в значительной степени ускоряет (в некоторых случаях до 6 раз [9]) сходимость

градиентного спуска по сравнению с вышеописанными функциями. Утверждается, что это связано с линейным характером функции и отсутствием насыщения.

Недостатки:

1. К сожалению, ReLU нейроны могут "умереть" во время обучения. Например, большой градиент, протекающий через нейрон ReLU приведет к тому, что веса обновятся таким образом, что этот нейрон никогда больше не будет активирован. Если это произойдет, то градиент, протекающий через данный нейрон начиная с этого момента всегда будет равен нулю. При правильной настройке скорости обучения эта проблема встречается достаточно редко.

Глава 2. Практическая часть

В данной главе будет рассмотрен процесс решения задач локализации и классификации дорожных знаков, и будет приведена реализация системы на платформе Android. Для решения задач использованы язык программирования Java, Android SDK, библиотека компьютерного зрения OpenCV 3.1.0 и фреймворк Caffe, предоставляющий методы глубокого обучения.

2.1. Локализация дорожных знаков

Чтобы распознавать дорожные знаки, первым делом необходимо уметь их локализовывать. Т.к. для достижения этой цели был выбран метод Виолы-Джонса, то нужно решить ряд подзадач:

1. Сформировать базу для обучения.
2. Обучить классификатор.

Рассмотрим реализацию этих подзадач и проанализируем результаты.

2.1.1. Формирование базы дорожных знаков

В свободном доступе нет базы российских дорожных знаков, которую можно использовать для обучения классификатора. Однако существуют базы других стран, которые используют для этих целей. Поэтому было решено взять за основу базу немецких знаков GTSRB [10], так как в ней большинство знаков внешне идентичны русским. База содержит знаки 43 видов и около 50000 их реальных изображений. Для использования в рамках работы база была отредактирована:

1. Из базы были удалены знаки, отсутствующие на российских дорогах, и знаки, распознавание которых в рамках данной работы рассмотрено не будет (рис. 9).

Рис. 9. Примеры удаленных из базы знаков

2. Добавлены некоторые отсутствующие знаки (3 вида): пешеходный переход, остановка запрещена, стоянка запрещена. Было взято по 12 реальных изображений каждого знака в качестве шаблона, после чего к ним были применены следующие преобразования (рис. 10):
    a. Искажение: шаблоны вращались под случайным углом.
    b. Яркость: шаблоны затемнялись или осветлялись на случайную величину [11].
    c. Расплывчатость: применялся *“motion”* фильтр [12] со случайными параметрами для создания эффекта, что изображение размылось при съемке.
    d. Шум: к шаблонам применялся случайный гауссов шум [13].

    В итоге, было создано по 900 искусственных экземпляров каждого дорожного знака.

Рис. 10. Последовательность преобразований при формировании базы знаков

Итого, из базы GTSRB было удалено 20 классов дорожных знаков и добавлено 3 новых класса. В новой базе получилось около 35000 изображений 26 классов дорожных знаков.

2.1.2. Обучение каскада классификаторов

Для увеличения точности локализации дорожных знаков было решено разделить их на 2 класса: треугольные (предупреждающие и “Пешеходный переход”), круглые (запрещающие), и обучить отдельные каскады классификаторов для каждого класса.

Для обучения каждого каскада была сформирована своя положительная и отрицательная выборка. Положительная выборка содержала только изображения со знаком (рис. 11а). Отрицательная выборка содержала изображения без знаков (рис. 11б): изображения улиц, природы, неба и т.д. От разнообразия негативной выборки зависит качество работы каскада в различных условиях. В итоге, размер каждой положительной выборки составил 1000 изображений, а отрицательной — 1500 изображений.

Рис. 11. Примеры из обучающей выборки круглых знаков:
а) положительные, б) отрицательные

Для обучения каскада классификаторов в пакете OpenCV есть программа *“opencv_traincascade.exe”* которая на вход принимает следующие данные:

- *data* — адрес папки, куда класть полученные результаты.

- *numStages* 18 — количество уровней каскада, которые программа будет обучать. Чем больше уровней, тем точнее, но дольше он работает. Нормальное их количество от 16 до 25.
- *minhitrate* 0,999 — коэффициент, определяющий качество обучения. По сути (1-0,999) — количество пропущенных объектов на слое.
- *numPos* 1000 — количество положительных образцов.
- *numNeg* 1500 — количество отрицательных образцов.

Каскады обучались по полтора дня и была получена следующая точность локализации (тестирование проводилось на тестовой базе GTSDB [14]) при коэффициенте увеличения сканирующего окна равного 1,1:

- треугольные знаки — 82,5%;
- круглые знаки — 74%.

2.1.3. Результаты

Обучено 2 каскада классификаторов, один — для треугольных знаков, второй — для круглых. Их точность при коэффициенте увеличения сканирующего окна, равному 1,1, составила 82,5% и 74% соответственно. Данная точность устраивает, потому что классификаторы работают с данными с камеры, и если объект не был найден на одном кадре, он может быть найден на следующем.

Было замечено, что большинство ошибок связано с маленьким размером дорожного знака (меньше 30×30). Пример удачной локализации представлен на рисунке 12а, неудачной — на рисунке 12б.

а) б)

Рис. 12. Примеры локализации дорожных знаков: а) удачный, б) неудачный

2.2. Классификация дорожных знаков

После того как дорожные знаки локализованы, необходимо их классифицировать. Т.к. для достижения этой цели были выбраны свёрточные нейронные сети, то нужно решить ряд подзадач:

1. Подготовить базу для обучения.
2. Обучить нейронную сеть.

Рассмотрим реализацию этих подзадач и проанализируем результаты.

2.2.1. Подготовка данных для обучения

Для обучения нейронной сети была использована база, созданная в параграфе 2.1.1. Она была разделена на обучающую и тестовую выборки. Обучающая выборка составила 27200 цветных изображений, а тестовая — 8200. Перед обучением сети изображения были предобработаны. Процесс предобработки:

1. *Вырезание знака.* Знаки были обрезаны по ограничивающему их квадрату.

2. *Преобразование в градации серого.* Для более быстрой работы нейронной сети принято решение использовать одну карту на входе, для этого все изображения были преобразованы в черно-белый вариант. Если бы данные были цветными, то на входе сети должно было быть три карты для каждого цветового канала, что привело бы к дополнительным вычислительным нагрузкам.
3. *Масштабирование.* Размеры знаков в базе варьируются от 15×15 до 250×250. Для использования свёрточных нейронных сетей необходимо, чтобы все входные данные были одного размера. Было необходимо подобрать такой размер, который бы удовлетворял следующим требованиям: не должно сильно снизиться исходное качество изображений; сеть должна работать довольно быстро. В результате произведенных исследований (табл. 1) было выявлено, что оптимальным размером входных данных будет 32×32. Масштабирование было применено методом бикубической интерполяции [15].
4. *Контрастное выравнивание.* Чтобы снизить влияние колебания контраста между изображениями на качество классификации было применено контрастное выравнивание CLAHE (рис. 13) (*Contrastlimited adaptive histogram equalization*) [16]. Этот метод улучшает контраст полутонового изображения путем преобразования значений его пикселей методом контрастно ограниченной адаптивной эквализации гистограммы. Он работает с небольшими областями изображения. Контраст каждой части изображения повышается, что связано с изменением формы гистограммы. После выполнения выравнивания (эквализации) метод объединяет края локальных областей с применением билинейной интерполяции, исключая искусственно созданные границы.

Табл. 1. Результаты экспериментов с размером входных данных

№	Размер входных данных	Точность	Время прямого прохода (мс)
1	40 × 40	98,69%	1,71
2	32 × 32	97,89%	0,9
3	24 × 24	94,77%	0,36

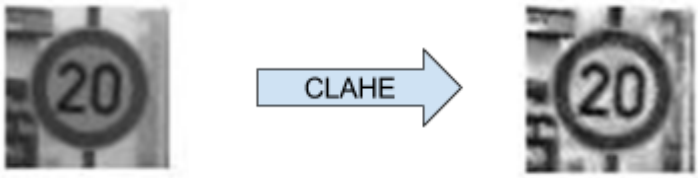

Рис. 13. Результат CLAHE выравнивания

2.2.2. Обучение нейронной сети

Для создания и обучения нейронной сети была использована библиотека машинного обучения Caffe [17]. Она написана на C++/Cuda и предоставляет MATLAB и Python интерфейсы.

Основной сложностью в работе с нейронными сетями является подбор параметров сети. За основу была взята типичная структура для свёрточных нейронных сетей со следующей последовательностью слоев: входной — свёрточный — субдискретизирующий — свёрточный — субдискретизирующий — полносвязный — полносвязный — полносвязный (выходной). Количество карт на слоях подбиралось экспериментально. Результаты экспериментов представлены в таблице 2. Для измерения времени и точности использовались предоставляемые библиотекой утилиты *“caffe time”* и *“caffe test”* соответственно. Время измерялось на двухъядерном процессоре Intel Core i5-3210M 2.50 GHz.

Табл. 2. Результаты экспериментов

№	Архитектура	Точность	Время прямого прохода (мс)
1	1-16-16-32-32-200-100-26	98,18%	1,86
2	1-12-12-24-24-150-100-26	97,89%	0,9
3	1-8-8-16-16-100-80-26	95,43%	0,45
4	1-6-6-12-12-80-60-26	90,12%	0,29

Все варианты обучались методом стохастического градиентного спуска [18] в течение 40 эпох. В качестве функции активации использовалась ReLU из-за ее преимуществ, описанных в параграфе 1.2.3.

Из рассмотренных архитектур была выбрана вторая, потому что скорость ее работы выше в 2 раза по сравнению с первой, а по точности она практически не уступает. Ее детальное описание представлено в таблице 3.

Табл. 3. Описание выбранной архитектуры

Слой	Тип	Количество карт и нейронов	Ядро
0	входной	1 карта 32×32 нейрона	
1	свёрточный	12 карт 28×28 нейронов	5×5
2	субдискретизирующий(max)	12 карт 14×14 нейронов	2×2
3	свёрточный	24 карты 10×10 нейронов	5×5
4	субдискретизирующий(max)	24 карты 5×5 нейронов	2×2
5	полносвязный	150 нейронов	
6	*“dropout”* слой (0,5)		
7	полносвязный	100 нейронов	
8	*“dropout”* слой (0,5)		
9	полносвязный	26 нейронов	
10	*“softmax”* слой		

Для снижения эффекта переобучения сети использовались “*dropout”* слои с вероятностью 0,5. Его суть заключается в том, что с заданной вероятностью нейроны сети отключаются и не участвуют в текущей итерации обучения. Также этот слой увеличивает обобщающие способности сети.

“Softmax” слой добавлен для понятной интерпретации выхода сети. Значения данного слоя находятся в интервале $(0, 1)$, и их можно рассматривать как вероятность принадлежности изображения соответствующему классу. Значения нейронов в нем определяются следующей формулой:

$$y_i = \frac{e^{z_i}}{\sum_{j=1}^{n} e^{z_j}},$$

где y_i — значение соответствующего нейрона *“softmax”* слоя, z_i — значение соответствующего нейрона предыдущего слоя.

2.2.3. Результаты

Была обучена свёрточная нейронная сеть, показывающая следующие результаты:

- Точность классификации — 97,89%, для сравнения, при обучении на контрастно невыровненных данных эта же сеть показала точность 94,12%.
- Время классификации одного предобработанного изображения — 0,9 мс на двухъядерном процессоре Intel Core i5-3210M 2.50 GHz.

Проанализированы ошибки классификации. В основном они связаны со следующими проблемами:

- плохое качество исходного изображения или темное изображение;
- маленький исходный размер знака на изображении;
- наличие бликов, смазанностей на изображении.

2.3. Реализация на платформе Android

Рассмотрим реализацию этапов локализации и классификации дорожных знаков на платформе Android.

2.3.1. Этап локализации

Библиотека компьютерного зрения OpenCV предоставляет функцию для использования уже обученного каскада *“detectMultiScale”* со следующими параметрами:

- *const Mat& image* — изображение, на котором будет проведен поиск объектов;
- *vector<Rect>&objects* — вектор, для хранения границ найденных объектов;
- *double scaleFactor* — коэффициент увеличения сканирующего окна;
- *int minNeighbors* — количество соседей. Этот коэффициент используется для того, чтобы каскад не выдавал в качестве ответа несколько рядом находящихся областей (на расстоянии нескольких пикселей);
- *Size minSize* — минимальный размер сканирующего окна;
- *Size maxSize* — максимальный размер сканирующего окна.

Так как OpenCV предоставляет интерфейс для платформы Android, можно воспользоваться этой функцией для локализации знака.

Т.к. дорожные знаки стоят на правой стороне дороги и расположены достаточно высоко, то не имеет смысла их искать на всем изображении с камеры телефона, можно сузить область до правой верхней части изображения (рис. 14). Это ускорит процесс локализации более чем в 2 раза.

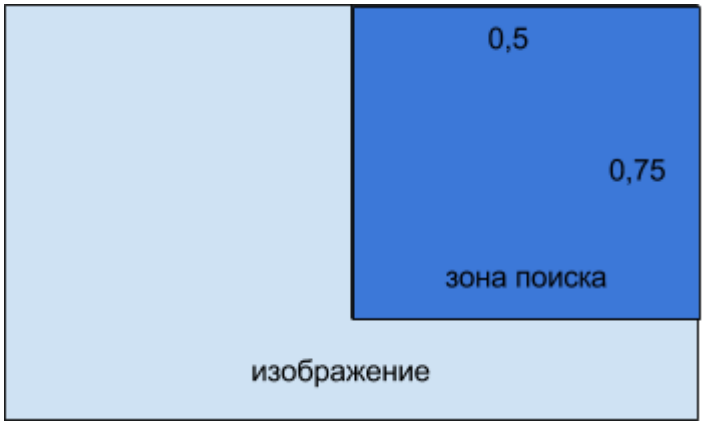

Рис. 14. Зона поиска на изображении с камеры

Время, необходимое для поиска дорожных знаков одним каскадом с использованием смартфона Asus ZenFone 2 (ZE551ML) на изображении размера 1280×720 при параметрах *scaleFactor* = 1,1; *minSize* = 30×30; *maxSize* = 70×70:

- на исходном изображении — 530-550 мс;
- в правой верхней части изображения — 160-170 мс.

Указанные параметры были выбраны в качестве значений по умолчанию. *ScaleFactor* фактор был подобран экспериментально, как компромиссный вариант между скоростью и точностью работы. Параметры *minSize* и *maxSize* пользователь может сам изменить в настройках программы.

Процесс локализации может быть описан следующей последовательностью шагов:

1. Получить изображение с камеры телефона.
2. Вырезать правую верхнюю часть как показано на рисунке 14.
3. Применить обученные каскадные классификаторы с помощью функции *“detectMultiScale”*. Для более быстрой обработки кадров и, как следствие, более плавной их смены в видео решено применять каскады не к одному и тому же кадру, а по очереди, т.е. один каскад — к первому кадру, второй — к следующему и т.д.
4. Запомнить и отрисовать полученные границы дорожных знаков.

Пример локализации представлен на рисунке 15.

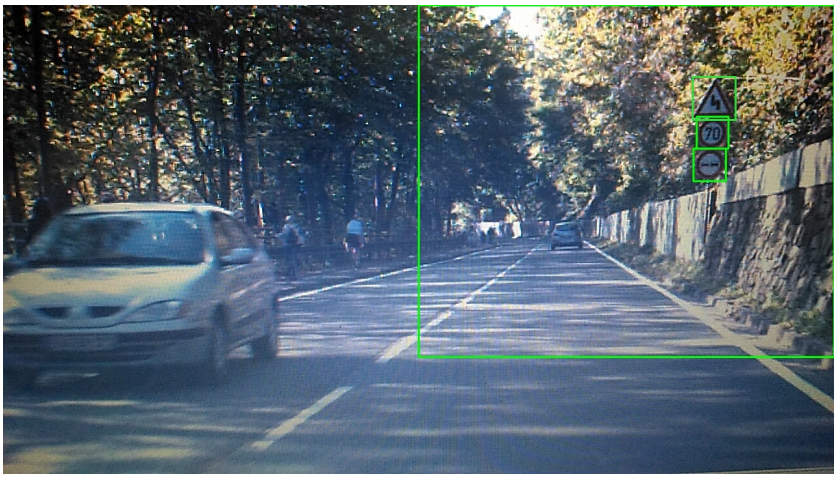

Рис. 15. Пример работы каскадов классификаторов на платформе Android

2.3.2. Этап классификации

Для того, чтобы работать с библиотекой Caffe на платформе Android был использован ее интерфейс для этой платформы Caffe-Android-Lib [19], требованием которого является версия Android выше 5.0.

Процесс классификации можно описать следующей последовательностью шагов:

1. Вырезать полученную после этапа локализации область изображения.
2. Преобразовать ее в оттенки серого.
3. Изменить размер области до размера 32×32 методом бикубической интерполяции.
4. Применить контрастное выравнивание CLAHE.
5. Применить обученную нейронную сеть.
6. Вывести результат.

Время, затрачиваемое на классификацию одного знака на смартфоне Asus Zenfone 2 (ZE551ML), составляет в среднем — 10 мс. Скриншоты программы представлены на рисунках 16 и 17.

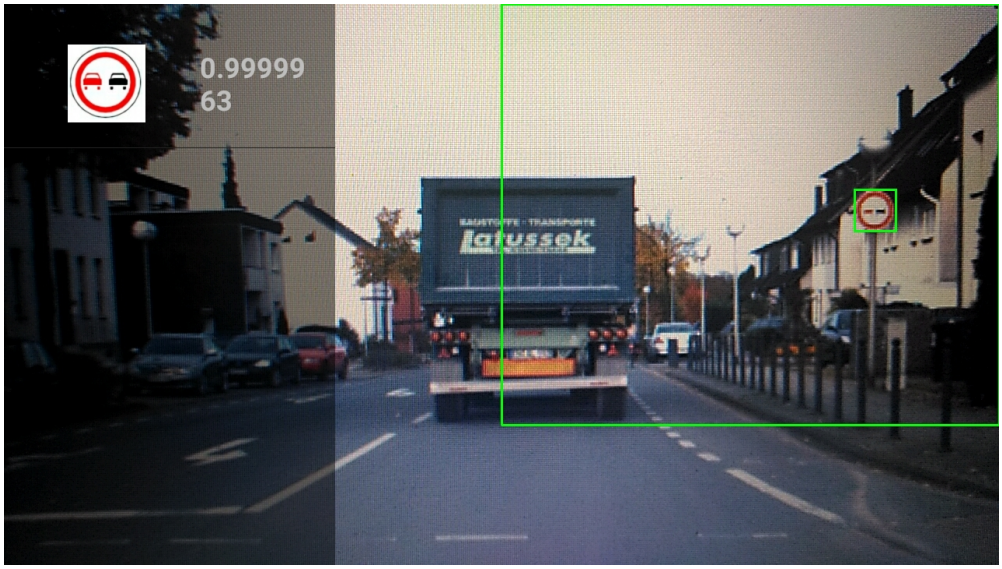

Рис. 16. Пример работы программы

Рис. 17. Пример работы программы

2.1.3. Результаты

Для того, чтобы оценить результаты работы системы на платформе Android в целом, было проведено ее тестирование на дорогах Петергофа. Система тестировалась в светлое время суток на телефоне Asus ZenFone 2 (ZE551ML). Были подсчитаны следующие данные: количество найденных знаков (верно локализованных), количество ненайденных знаков, количество ложных срабатываний на этапе локализации

(объект, не являющийся знаком, детектировался как знак), количество верно классифицированных, количество неверно классифицированных. Всего на пути следования было 55 знаков. Рассматривались только знаки, присутствующие в обучающей базе. Результаты представлены в таблице 4.

Табл. 4. Результаты тестирования

Детектировано (точность локализации)	Не детектировано	Ложное детектирование	Верно классифицировано (точность классификации)	Неверно классифицировано
49 (89,09%)	6	3	38 (77,55%)	11

$$\text{точность локализации} = \frac{\text{кол-во детектированных}}{\text{общее число знаков}}$$

$$\text{точность классификации} = \frac{\text{кол-во верно классифицированных}}{\text{кол-во детектированных}}$$

Среднее время, затрачиваемое на полную обработку одного кадра (локализация и классификация) на смартфоне Asus Zenfone 2 (ZE551ML), составляет — 190-200 мс, что примерно соответствует частоте обновления кадров 5 fps.

Выводы

В данной работе были достигнуты следующие результаты:

- На этапе локализации дорожных знаков обучено два каскада классификаторов: один — для круглых знаков (запрещающих), второй — для треугольных (предупреждающих и знака “Пешеходный переход”). Точность их работы 82,5% и 74% соответственно. Т.к. классификаторы работают с изображениями, полученными с камеры, то если объект не будет найден на одном кадре, он может быть найден на следующем, поэтому данная точность приемлема. Среднее время, необходимое одному каскаду для обработки изображения размера 1280×720 на смартфоне Asus ZenFone 2 (ZE551ML) составляет 530-550 мс. Учитывая, что вычисления рассматриваются на мобильной платформе, указанное время удовлетворительное, однако для работы в режиме реального времени этого недостаточно. Поэтому было необходимо оптимизировать данный метод указанием зоны поиска на изображении (рис. 14), что сократило время до 160-170 мс.
- На этапе классификации дорожных знаков была выбрана структура свёрточной нейронной сети, дающая лучший результат с точки зрения соотношения точности и скорости работы. Ее точность — 97,89%, а время, необходимое на распознавание одного знака на смартфоне Asus ZenFone 2 (ZE551ML) — 10 мс. Данные показатели достаточно высокие и позволяют успешно использовать сеть в режиме реального времени.
- При тестировании системы целиком на дорогах Петергофа с использованием смартфона Asus ZenFone 2 (ZE551ML) получены следующие показатели: точность детектирования — 89,09%, точность классификации — 77,55%, время на обработку одного кадра —

190-200 мс (примерно 5 fps). Невысокая точность классификации объясняется следующими причинами:

a. Камера. Хотя сейчас телефоны оснащаются довольно мощными камерами с высоким разрешением, у них есть проблемы со съемкой в движении и автофокусом, из-за чего изображение часто размывается и система допускает ошибки.
b. Немецкие знаки в обучающей выборке. Сеть обучалась на немецких знаках и показала точность 97,89% на немецких знаках. Русские же знаки, хоть и схожи с немецкими, но немного отличаются, поэтому и результат ниже. Для более высокого показателя точности необходима база русских дорожных знаков.

Касательно времени обработки кадра можно заметить, что большую часть времени обработки кадра занимает этап локализации (примерно 80%). Поэтому для увеличения скорости работы системы нужна дополнительная оптимизация алгоритма локализации. Например, можно анализировать цветовые каналы изображения (для запрещающих и предупреждающих знаков особый интерес представляет красный канал) и находить “зоны интереса”, то есть области изображения, в которых могут находиться знаки. Этот подход может существенно уменьшить область поиска для каскадов и, как следствие, ускорить процесс локализации.

Заключение

В ходе исследования разработана система автоматического распознавания запрещающих и предупреждающих дорожных знаков и знака “Пешеходный переход” на платформе Android:

1. На первом этапе подготовлена база дорожных знаков и решена задача локализации знаков методом Виолы-Джонса. Полученное решение показало достаточно высокую точность локализации.
2. На втором этапе обучена свёрточная нейронная сеть для решения задачи классификации выделенных дорожных знаков. Подобрана архитектура сети с высоким показателем точности и скорости классификации.
3. На третьем этапе представлена реализация системы на платформе Android и приведены приемы, увеличивающие скорость работы системы. Используемые в работе алгоритмы носят общий характер и поэтому могут быть легко перенесены и на другие платформы.

Реализацию системы, каскадные классификаторы и обученную нейронную сеть можно увидеть в приложении [20].

Список литературы

1. Емкость парка легковых автомобилей в России на 2016 год. http://avtostat-info.com/News/2174
2. Traffic sign recognition on Wikipedia. https://en.wikipedia.org/wiki/Traffic_sign_recognition
3. Viola P.A., Jones M.J. Robust real-time face detection // International journal of computer vision, 2004. Vol. 57, No 2, P. 137-154.
4. LeCun Y., Bengio Y. Word-level training of a handwritten word recognizer based on convolutional neural networks // Proceedings of the International Conference on Pattern Recognition, Jerusalem, Israel, 1994. Vol. 2, P. 88-92.
5. Lawrence S., Giles C.L., Tsoi A.C., Back A.D. Face recognition: a convolutional neural network approach // IEEE Transactions on Neural Networks, 1997. Vol. 8, No 1, P. 98-113.
6. Хайкин С. Нейронные сети: полный курс, 2 изд. М.: Вильямс, 2008. 1103 с.
7. Осовский С. Нейронные сети для обработки информации. М.: Финансы и статистика, 2002. 344 с.
8. Must know tips/tricks in deep neural networks. http://lamda.nju.edu.cn/weixs/project/CNNTricks/CNNTricks.html
9. Krizhevsky A., Sutskever I., Hinton G.E. Imagenet classification with deep convolutional neural networks // Advances in neural information processing systems 25, 2012. P. 1097-1105.
10. The german traffic sign recognition benchmark (GTSRB) dataset. http://benchmark.ini.rub.de/?section=gtsrb&subsection=dataset
11. Image Processing Toolbox: управление яркостью палитры. http://matlab.exponenta.ru/imageprocess/book3/10/brighten.php

12. Image Processing Toolbox: create predefined 2-D filter. http://www.mathworks.com/help/images/ref/fspecial.html
13. Image Processing Toolbox: добавление шума. http://matlab.exponenta.ru/imageprocess/book3/10/imnoise.php
14. The german traffic sign detection benchmark (GTSDB) dataset. http://benchmark.ini.rub.de/?section=gtsdb&subsection=dataset
15. Бикубическая интерполяция. http://ru.wikipedia.org/?oldid=68800582
16. Contrastlimited adaptive histogram equalization (CLAHE). http://www.mathworks.com/help/images/ref/adapthisteq.html
17. Caffe. http://caffe.berkeleyvision.org
18. Stochastic gradient descent in Caffe. http://caffe.berkeleyvision.org/tutorial/solver.html
19. Интерфейс библиотеки Caffe для Android. https://github.com/sh1r0/caffe-android-lib
20. Реализация системы автоматического распознавания знаков на платформе Android. https://github.com/Nazukin-Dmitry/TrafficSignsRecognition